Lab5:

## Q1:

**Write a single server and multiple client program to illustrate multiple clients communicating with a concurrent server. The client1 on establishing successful connection sends ”Institute Of” string to the server along with its socket address. The client2 on establishing successful connection sends ”Technology” string to the server along with its socket address. The server opens a text file having the keyword ”Manipal”, append the keywords ”Institute of” and ”Technology” and displays” Manipal Institute of Technology” along with the socket addresses of the clients . If the number of clients connected exceeds 2, the server sends ”terminate session” to all clients and the program terminates.**

Server:

#include<string.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<stdio.h>

#define MAXLINE 4096 /\*max text line length\*/

#define SERV\_PORT 4444 /\*port\*/

#define LISTENQ 8 /\*maximum number of client connections\*/

int main()

{

int s,r,recb,sntb,x,ns,a=0;

printf("INPUT port number: ");

scanf("%d", &x);

socklen\_t len;

struct sockaddr\_in server,client;

char buff[100];

char str[100];

s=socket(AF\_INET,SOCK\_STREAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

server.sin\_addr.s\_addr=htonl(INADDR\_ANY);

r=bind(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nBinding error.");

exit(0);

}

printf("\nSocket binded.");

r=listen(s,LISTENQ);

if(r==-1)

{

close(s);

exit(0);

}

printf("\nSocket listening.\n");

int childpid,n,no\_of\_clients=0;

for(;;){

len=sizeof(client);

ns=accept(s,(struct sockaddr\*)&client, &len);

if(ns==-1)

{

close(s);

exit(0);

}

printf("\nSocket accepting.\n");

if(no\_of\_clients==2)

{

printf("\nMore than 2 clients!\n");

close(ns);

break;

}

no\_of\_clients++;

if ( (childpid = fork ()) == 0 )

{//if it’s 0, it’s child process

//printf ("%s\n","Child created for dealing with client requests");

//close listening socket

close (s);

n = recv(ns, buff, sizeof(buff),0);

//strcat(str,buff);

if (n < 0){

printf("%s\n", "Read error");

exit(0);

}

FILE \*fptr;

fptr=fopen("tmp.txt","a");

fputs(buff,fptr);

fclose(fptr);

char buff2[50];

n = recv(ns, buff2, sizeof(buff2),0);

if (n < 0){

printf("%s\n", "Read error");

exit(0);

}

FILE \*fptr2;

fptr2=fopen("tmp2.txt","a");

fputs(buff2,fptr2);

fclose(fptr2);

if(no\_of\_clients==2)

{

FILE \*fp;

int line\_num = 1;

char temp[512];

if((fp = fopen("tmp.txt", "r")) == NULL) {

close(s);

exit(0);

}

while(fgets(temp, 512, fp) != NULL) {

strcpy(str,temp);

printf("%s", str);

line\_num++;

}

if(fp) {

fclose(fp);

}

FILE \*fp2;

int line\_num2 = 1;

char temp2[512];

if((fp2 = fopen("tmp2.txt", "r")) == NULL) {

close(s);

exit(0);

}

while(fgets(temp2, 512, fp2) != NULL) {

strcpy(str,temp2);

printf("\n%s", str);

line\_num2++;

}

if(fp2) {

fclose(fp2);

}

}

exit(0);

}

//close socket of the server

close(ns);

}

}

Client1:

#include<string.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<netdb.h>

#include<errno.h>

int main()

{

int s,r,recb,sntb,x;

printf("INPUT port number: ");

scanf("%d", &x);

struct sockaddr\_in server;

char buff[50];

s=socket(AF\_INET,SOCK\_STREAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

r=connect(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nConnection error.");

exit(0);

}

printf("\nSocket connected.");

strcpy(buff,"Institute Of");

sntb=send(s,buff,sizeof(buff),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

struct hostent \*host\_entry;

int hostname;

char hostbuffer[256];

char \*ipbuffer;

hostname=gethostname(hostbuffer,sizeof(hostbuffer));

host\_entry=gethostbyname(hostbuffer);

ipbuffer = inet\_ntoa(\*((struct in\_addr\*)host\_entry->h\_addr\_list[0]));

char ip[50];

strcpy(ip,ipbuffer);

strcat(ip," ");

printf("\nIP is: %s",ip);

sntb=send(s,ip,sizeof(ip),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

close(s);

}

Client2:

#include<string.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<netdb.h>

#include<errno.h>

int main()

{

int s,r,recb,sntb,x;

printf("INPUT port number: ");

scanf("%d", &x);

struct sockaddr\_in server;

char buff[50];

s=socket(AF\_INET,SOCK\_STREAM,0);

if(s==-1)

{

printf("\nSocket creation error.");

exit(0);

}

printf("\nSocket created.");

server.sin\_family=AF\_INET;

server.sin\_port=htons(x);

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

r=connect(s,(struct sockaddr\*)&server,sizeof(server));

if(r==-1)

{

printf("\nConnection error.");

exit(0);

}

printf("\nSocket connected.");

strcpy(buff," Technology");

sntb=send(s,buff,sizeof(buff),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

exit(0);

}

struct hostent \*host\_entry;

int hostname;

char hostbuffer[256];

char \*ipbuffer;

hostname=gethostname(hostbuffer,sizeof(hostbuffer));

host\_entry=gethostbyname(hostbuffer);

ipbuffer = inet\_ntoa(\*((struct in\_addr\*)host\_entry

>h\_addr\_list[0]));

char ip[50];

strcpy(ip,ipbuffer);

strcat(ip," ");

printf("\nIP is: %s",ip);

sntb=send(s,ip,sizeof(ip),0);

if(sntb==-1)

{

close(s);

printf("\nMessage Sending Failed");

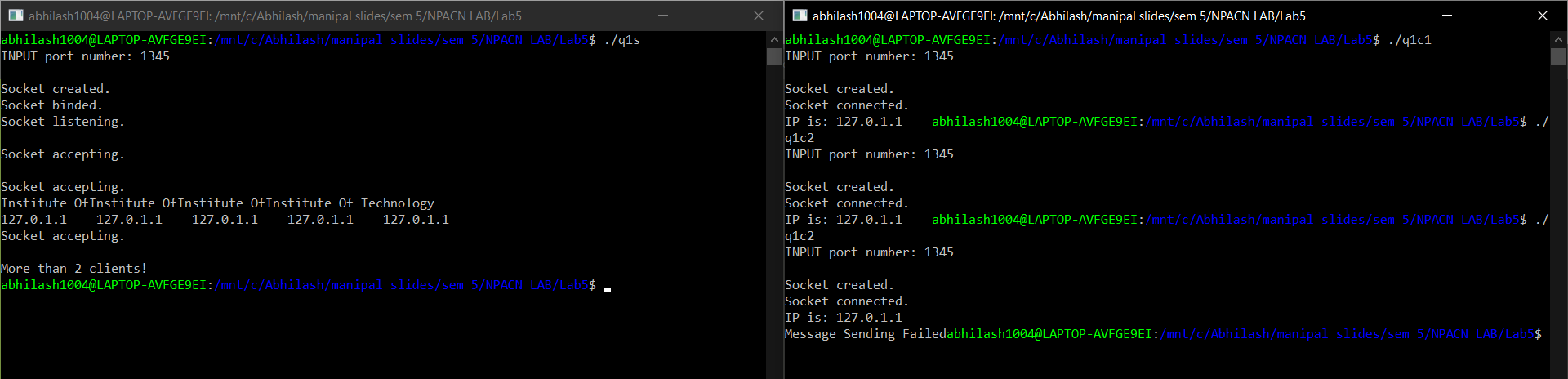
exit(0);

}

close(s);

}

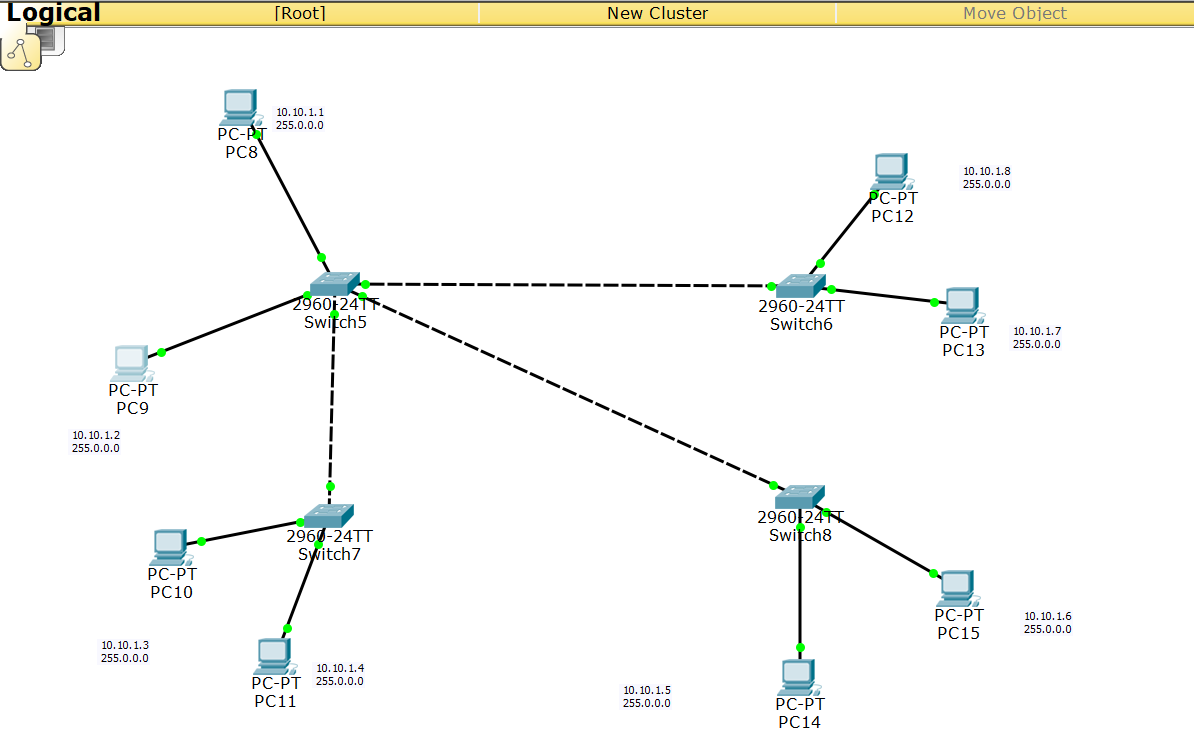
Output:

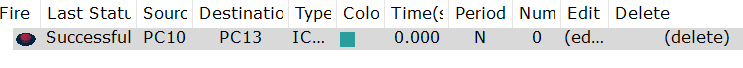


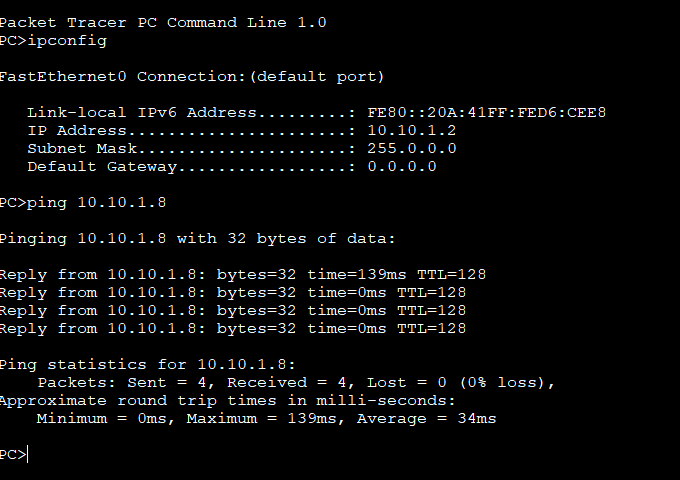
Lab8:

Q1:

Set up a network with 4 switches and 8 PC’s and verify the connectivity between all the PC’s.

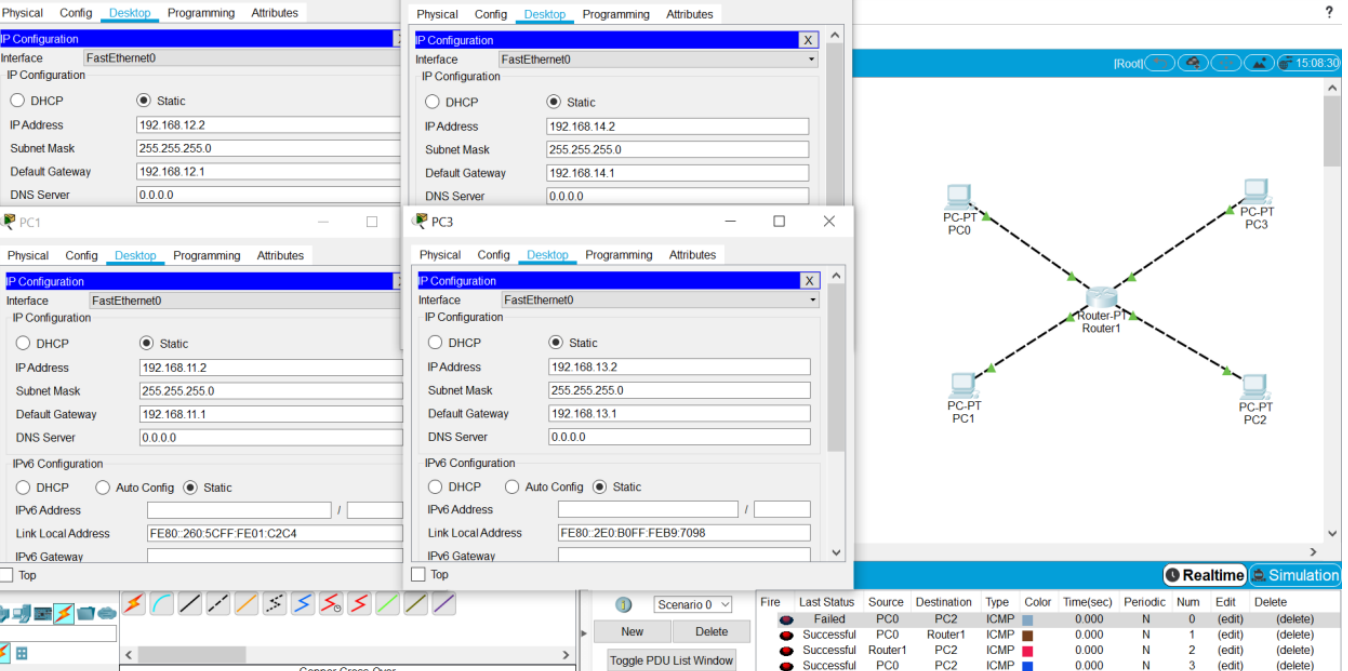


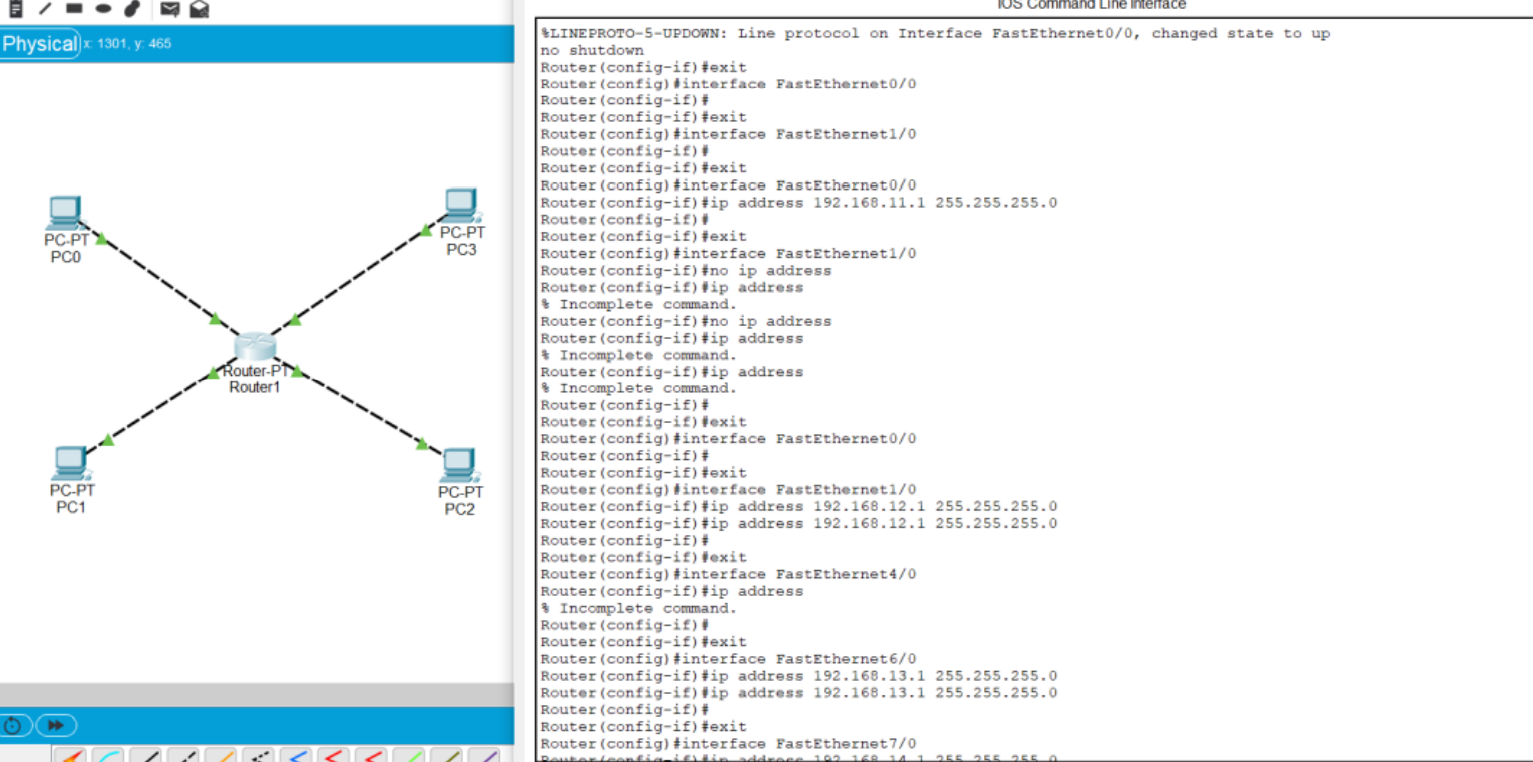


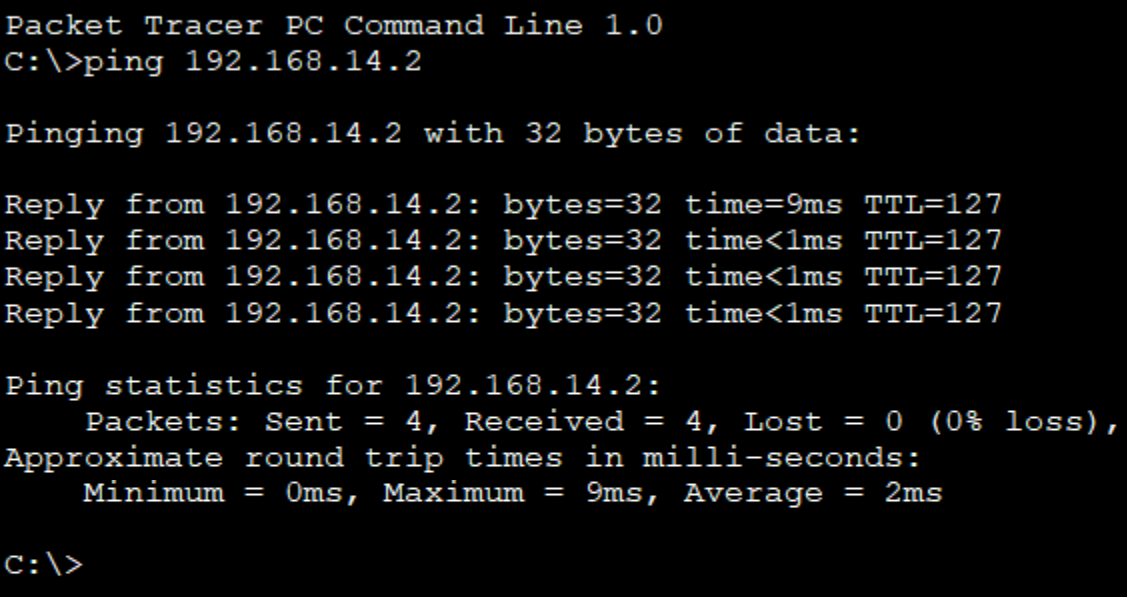


Q2:

Design and configure a network using 4 PC’s and a router. Configuration should be done using Command Line Interface (CLI).







Lab9:

adq1:

